

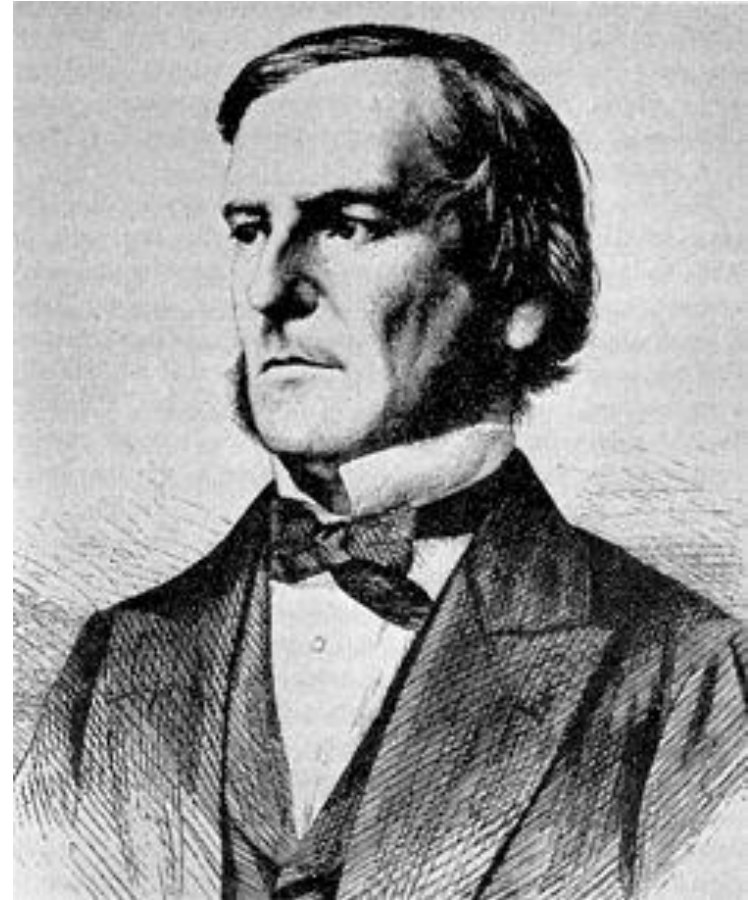
9. Логически данни

Проф. д-р Емил Хаджиколев

1. Логически съждения;
2. Булева алгебра;
3. Функции на една и две булеви променливи;
4. Логически оператори;
5. Логически изрази;
6. Условен оператор.

Булева алгебра

- Джордж Бул, английски учител по математика, 1847г.
- Опит за формализиране на логиката с цел въвеждане и използване на алгебрични методи в логиката.



Логически съждения

- Основен обект на изследване в булевата алгебра са **логическите съждения**.
- **Съждение** - твърдение, което може да се оцени като **истина** (true, 1) или **лъжа** (false, 0).
- **Примери:**
 - „България е държава” е съждение, което има стойност истина.
 - „България е континент” е съждение, със стойност лъжа.
 - „Къде се намира България?” и „Само ако можех да спечеля 1 милион лева от тотото!” не са съждения.

Предикати

- **Предикат** - съждение, което в определен момент не може да бъде оценено като истина или лъжа.
- **Пример за предикат:** „Числото x е четно”
 - Ние не знаем кое е числото x , затова не можем да определим дали е четно или нечетно.
 - При зададена конкретна стойност на променливата x , напр. $x = 20$, изречението „Числото 20 е четно” става съждение, което може да се оцени като вярно.

Прости и съставни съждения

- **Просто съждение** - съждение, което се формулира чрез просто изречение и утвърждава или отрича един признак.
 - **Пример:** „Земята е кръгла”
- **Съставно съждение** – състои се от две или повече прости съждения. Конструира се с помощта на логически съюзи и словосъчетания – „не”, „и”, „или”, „ако ... то”, „тогава и само тогава, когато” и т.н.
 - **Пример:** „Земята е кръгла и Луната се върти около Земята”

Логически променливи и функции

- **Логически константи** – 0 и 1.
- **Логическа променлива** - величина, която може да приема различни стойности във времето – 0 или 1.
- **Логическа (булева) функция** - функция, чиято стойност зависи от краен брой логически променливи.
 - Множеството от конкретни стойности на променливите на дадена логическа функция се нарича **набор**.
 - Една логическа функция може да се дефинира по няколко начина - **словесно** (чрез текст), **таблично** (чрез таблици на истинност), **аналитично** (чрез формула) или **графично** (чрез графика).
 - **Табличното представяне** на булева функция изисква явно задаване на всевъзможните набори от допустими стойности на променливите, и съответната стойност на функцията.

Функции на една булева променлива

- Брой на възможните набори – два;
- Брой на функциите – четири.
 - **Идентитетът** е функция, която повтаря стойностите на променливата величина.
 - При **отрицанието**, функцията приема стойности, противоположни на стойностите на променливата.
 - Функция, която има стойност 0 („невярно“), без значение от конкретната стойност на променливата, се нарича **противоречие**
 - Ако стойността на една функция е винаги 1 („вярно“), функцията се нарича **тавтология**.

р	$F_1(p)$ Идентитет	$F_2(p)$ Отрицание	$F_3(p)$ Противоречие	$F_4(p)$ Тавтология
0	0	1	0	1
1	1	0	0	1

Логика на основните функции (1)

- **Конюнкция (логическо И)** – резултатът е истина, само ако и двата операнда (p и q) са истина; в противен случай е лъжа.
- **Дизюнкция (логическо ИЛИ)** – резултатът е истина, ако поне единия от двата операнда (p или q) е истина; в противен случай е лъжа.
- **Импликация (следствие)** – има стойност истина в два случая – ако от „лъжа следва лъжа или истина” и „от истина следва истина”; лъжа е само случая, че „от истина следва лъжа”.
- **Еквивалентност** – резултатът е истина, ако двата операнда (p и q) имат една и съща стойност; в противен случай е лъжа.

Логика на основните функции (2)

- **Сума по модул 2 (изключващо или)** – резултатът е истина, ако двата операнда (p и q) са различни; иначе е лъжа. Друга логика (в контекста на „изключващо или“) е: резултатът е истина, ако само първия или само втория операнд са истина; в противен случай е лъжа. Трети вариант за интерпретация, свързан със „сума по модул 2“ е, че функцията е подобна на побитово сумиране – $0+0=0$; $0+1=1$; $1+0=1$; $1+1=(1)0$. В резултата на последното равенство остава само нулата, а единицата преминава в по-старшия разряд.
- **Стрелка на Пирс (логическо ИЛИ-НЕ)** – отрицание на дизюнкцията.
- **Щрих на Шефер (логическо и-не)** – отрицание на конюнкцията.

Функционално пълна система от логически функции

- Някои логически функции могат да се изразят чрез други логически функции.
- **Функционално пълна система от логически функции** се нарича всяка съвкупност от краен брой логически функции, чрез които могат да се представят всички останали логически функции.
- **Примери** за функционално пълни системи от логически функции:
 - отрицание и конюнкция;
 - отрицание и дизюнкция;
 - и др.

Класически базис

- Най-голямо приложение в компютрите и езиците за програмиране намира т. нар. **класически базис**.
- **Класически базис** – състои се от функциите **конюнкция, дизюнкция и отрицание**.
- По-рядко използван, но зададен като възможен за използване оператор в езиците за програмиране е „**Исключващо ИЛИ**”.
- Тези четири оператора (функции) не са минимален базис, но са естествени и удобни за използване от хората.

Класически базис - означения

- Има различни математически означения за операторите от класическия базис.
- в различни езици за програмиране често се използват следните означения:

Функция/ езици	Отрицание	Логическо И	Логическо ИЛИ	Изключващо ИЛИ
В повечето ЕП	!	&&		^
В някои ЕП	NOT	AND	OR	XOR

Съждително смятане

- **Съждителен израз** е съвкупност от съждителни променливи a, b, c и т.н., свързани със знаци за логически операции (!, &&, || и др.) и скоби, за указване приоритета на операциите.
- От съществено значение е да се прави **разлика** между **съждение** и **съждителен израз**.
- **Съждителният израз** описва цял клас от съждения със сходна структура!

Съждително смятане - пример

- Съждителният израз $p \mid \mid q$ изразява логическата структура на следните съждения:
 - „Едно изречение е просто **или** едно изречение е съставно”
 - „Утре времето ще е топло **или** утре ще вали сняг“.
- Замествайки променливите в един съждителен израз със съждения, се получава съждение.
- За да се определи верността на едно сложно съждение, е необходимо да се знае каква е верността на съставлящите го прости съждения и смисъла на свързващите ги логически операции.

Приоритет на логическите оператори

- Логическите оператори си имат **приоритет**.
- **Приоритетът определя последователността на изпълнение на операторите.**
- **С най-висок приоритет** е отрицанието, следван от конюнкция, дизюнкция, импликация и еквивалентност.
- **За промяна на приоритета на операторите** се използват скоби.

Закони на съждителното смятане

- **Комутативен закон**

$$p \&\& q \Leftrightarrow q \&\& p$$

$$p \mid\mid q \Leftrightarrow q \mid\mid p$$

- **Асоциативен закон**

$$(p \&\& q) \&\& r \Leftrightarrow p \&\& (q \&\& r)$$

$$(p \mid\mid q) \mid\mid r \Leftrightarrow p \mid\mid (q \mid\mid r)$$

- **Дистрибутивен закон**

$$(p \mid\mid q) \&\& r \Leftrightarrow p \&\& r \mid\mid q \&\& r$$

$$(p \&\& q) \mid\mid r \Leftrightarrow (p \mid\mid r) \&\& (q \mid\mid r)$$

- **Закон за двойното отрицание**

$$!(\neg p) \Leftrightarrow p$$

- **Закони на де Морган**

$$\neg(p \&\& q) \Leftrightarrow \neg p \mid\mid \neg q$$

$$\neg(p \mid\mid q) \Leftrightarrow \neg p \&\& \neg q$$

- **Закон за контрапозицията**

$$p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$$

- **Закон за изключеното трето**

$$p \mid\mid \neg p \Leftrightarrow 1$$

- **Закон за транзитивност**

$$(p \rightarrow q) \&\& (q \rightarrow r) \Leftrightarrow p \rightarrow r$$

Доказателство на законите за съждителното смятане

- Законите за съждителното смятане могат лесно да бъдат доказани чрез използване на таблици за истинност, по следния начин:
 - Първо е необходимо да се образуват всички възможни набори от стойности на двете логически променливи p и q .
 - След това последователно се изчисляват и сравняват стойностите на левия и десния израз.
 - Ако двата израза имат равни стойности за еднаквите стойности на променливите, можем да твърдим че законът е общовалиден.

Пример: Да се докаже законът на де
Морган $!(p \ \&\& \ q) \Leftrightarrow !p \ || \ !q$

p	q	p && q	!(p && q)	!p	!q	!p !q
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Следствия от законите за съждителното смятане

- **Закон за сцепване**

$$p \ \&\& \ q \ || \ p \ \&\& \ !q \Leftrightarrow p$$

$$(p \ || \ q) \ \&\& \ (p \ || \ !q) \Leftrightarrow p$$

- **Закон за поглъщане**

$$p \ || \ (p \ \&\& \ q) \Leftrightarrow p$$

$$p \ \&\& \ (p \ || \ q) \Leftrightarrow p$$

- **Закон за съкращаване**

$$p \ || \ (!p \ \&\& \ q) \Leftrightarrow p \ || \ q$$

$$p \ \&\& \ (!p \ || \ q) \Leftrightarrow p \ \&\& \ q$$

Преобразуване на сложни изрази до по-прости

- Законите на съждителното смятане и следствията могат да се използват за преобразуване на сложните изрази до по-прости.
- **Пример:** Да се опрости изразът $p \mid\mid (p \&\& q)$.

$p \mid\mid (p \&\& q) =$

$(p \&\& 1) \mid\mid (p \&\& q) =$

$p \&\& (1 \mid\mid q) =$

$p \&\& 1 =$

p

Булев тип и литерали в C++

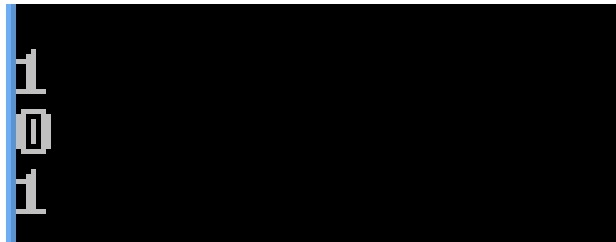
- Възможните стойности на **булевия тип bool** са **литералите true (истина) и false (лъжа)**.
- В C++ всеки израз със стойност 0 се приема за false, и всеки израз с различна от 0 стойност се преобразува до true (т.е. 1).

```
bool b;
```

```
b = true;  
cout << b << '\n'; // 1
```

```
b = false;  
cout << b << '\n'; // 0
```

```
b = 50;  
cout << b << '\n'; // 1
```



Логически оператори

Операндите на логическите оператори са логически стойности (изрази, които имат логическа стойност), резултатът също е логическа стойност.

Оператор	Име
& &	Логическо И
	Логическо ИЛИ
!	Отрицание – НЕ
^	Изключващо ИЛИ

Таблица на истинност на логическо И

a	b	a && b
false	false	false
false	true	false
true	false	false
true	true	true

Пример за логическо И:

- *До участие в конкурс се допускат лица, които са навършили 16 години и представят автобиография (CV).*
- За да участва конкретен човек (Иван) в конкурса трябва да удовлетворява едновременно и двете условия.

Таблица на истинност на логическо ИЛИ

a	b	a b
false	false	false
false	true	true
true	false	true
true	true	true

Пример за логическо ИЛИ:

- *За участие в състезание се допускат лица, които са навършили 12 години или са с придружител.*
- Достатъчно е само едно от условията да е изпълнено, за да се допусне Иван до конкурса:
 - да е по-голям от 12 г.;
 - да е с придружител;
 - може да са изпълнени и двете условия едновременно – по-голям от 12 г. и с придружител.

Таблица на истинност на логическо отрицание

a	!a
false	true
true	false

Пример за логическо отрицание:

- *За участие в конкурс се допускат лица, които не са по-малки от 12 г.*
- Отричаме (не допускаме) участието на хора по-малки от 12 г.
- Понякога може да се избегне отрицание: В примера, може да се изисква лицата да са по-големи от 12 г. т.е. ***!($<12г.$) е равносилно на ($\geq 12г.$).***

Таблица на истинност на изключващо ИЛИ

a	b	$a \wedge b$
false	false	false
false	true	true
true	false	true
true	true	false

Логически изрази

- В логическият израз участват един или повече логически оператори.
- Всеки от операторите има различен приоритет – първо се прилага логическото отрицание, след това И, и накрая ИЛИ.
- Ако искаме да променяме приоритета, използваме скоби, изразите в които се изчисляват с най голям приоритет.

- Еквивалентни, например са изразите

`!a && b || c` и `((!a) && b) || c`,

но съвсем друг смисъл имат напр.,

`!(a && b || c)`, `!(a && b) || c`, `!a && (b || c)`

- Логическите изрази се изчисляват отляво надясно, и може в процеса на изпълнение да не се изчислят докрай. Напр. няма смисъл да се изчислява под-израза в скобите ако е пресметнато, че първата част е `true` (може да не е литерал)

`true || (a || b && c...)`

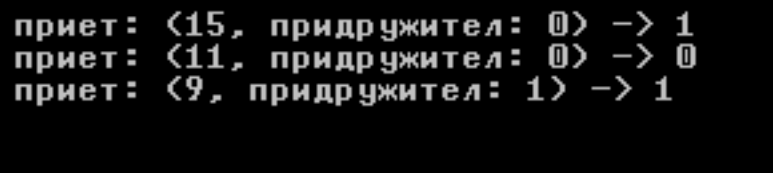
Ако първата част е `false`, ще се наложи да се изчисли и останалата част от израза.

Пример

```
#include <iostream>
#include <string>
using namespace std;

// Функция за оценяване дали участник отговаря на условията за участие:
// За участие в състезание се допускат лица, които са навършили 12 години или са с придружител.
// Параметри са възраст и "има ли придружител?".
void accept(int age, bool have_companion){
    bool accept = age > 12 || have_companion;
    cout << " приет: (" << age << ", придружител: " << have_companion << ") -> " << accept << '\n';
}

int main(){
    setlocale(LC_ALL, "bg");
    int age = 15; // задаване на стойности за участник
    bool have_companion = false;
    accept(age, have_companion); // проверка
    // може и без променливи в основната програма
    accept(11, false);
    accept(9, true);
    return 0;
}
```



```
приет: <15, придружител: 0> -> 1
приет: <11, придружител: 0> -> 0
приет: <9, придружител: 1> -> 1
```

Условен оператор ?:

- Чрез тройния условен оператор ?: се изчислява един от два възможни изрази, в зависимост от истинността на някакво логическо условие.
- **Синтаксисът** на тройния оператор е следният:
<логическо условие> ? <израз_true> : <израз_false>;
- Първият операнд е логическо условие. Другите два операнда са изрази, чийто резултатен тип обикновено е един и същ (но това не е задължително). Ако логическото условие има стойност true, оператора връща като резултат стойността на израза след ?, иначе – израза след :.
- Когато резултатът от изпълнението на условния оператор се присвоява на променлива, двата изрази трябва са от тип, съвместим с типа на променливата.

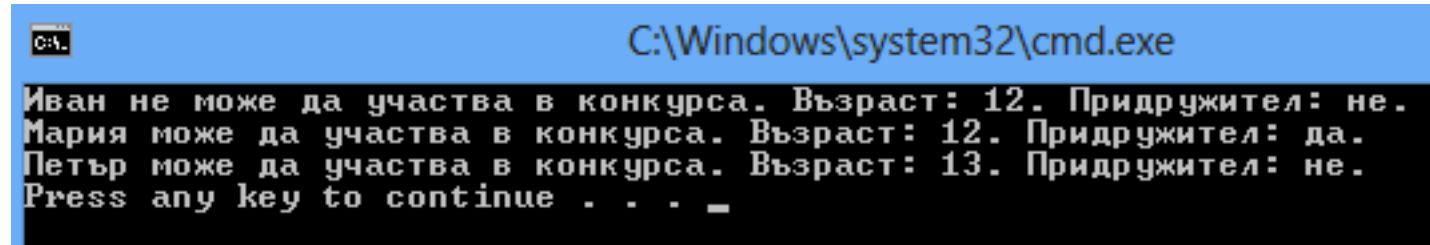
Подобрение на примера (с условен оператор)

```
#include <iostream>
#include <string>
using namespace std;

// Функция за оценяване дали участник отговаря на условията за участие:
// За участие в състезание се допускат лица, които са навършили 12 години или са с придружител.
// Параметри са име, възраст, "има ли придружител?".

void testPerson(string name, int age, bool have_companion) {
    bool accept = age > 12 || have_companion;
    string info = name + (accept ? "" : " не") + " може да участва в конкурса. Възраст: "
        + to_string(age) + ". Придружител: " + (have_companion ? "да" : "не") + ".";
    cout << info << '\n';
}

int main() {
    setlocale(LC_ALL, "bg");
    testPerson("Иван", 12, false);
    testPerson("Мария", 12, true);
    testPerson("Петър", 13, false);
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
Иван не може да участва в конкурса. Възраст: 12. Придружител: не.
Мария може да участва в конкурса. Възраст: 12. Придружител: да.
Петър може да участва в конкурса. Възраст: 13. Придружител: не.
Press any key to continue . . . _
```